**Task 1 (30 minutes)**

Take a look at the picture below and come up with suggestions on how this design could adapt to different devices (mobile, tablets etc). Things we're looking for is how the layout should change and look for mobile users, basic UX things to keep in mind and other best practices when developing responsive websites.

**UX Design suggestions**

* There are 3 menus on the page, top horizontal menu, left vertical menu and a breadcrumb on the top. All these 3 can be be replaced with a mobile-first UX design provided by Bootstrap called as Hamburger menu. In addition to it, the breadcrumbs can then be replaced by just the page name that they are in.
  + This type of hamburger design is stuiable for mobile/iPad functionalities
* The footer shows ability to view children and etc, this heirarchy of pages can be added in the Hamburger menu as I described above with down arrows to collapse and expand the menus.
* Can also add "plus" on the top of the page list in the menu allowing the user to "Add a new page" this getting rid of the footer totally.
* Can move the Comments bar on the bottom of the page to a vertical division on the right side of the page that occupies just 15% of the page width with scroll feature that functions independently of the main page scroll feature.

Thus the user can view comments simultaneously when the page has data on the left.

* When opened in a mobile or iPad the comments can scroll to the bottom of the page

* Can remove the "Search" button as the search bar is self-explanatory.
* Move the feature to "Share" and "Flag" to the main menu bar. That might give more room for data on the page.
* Need to have more features for ability to add "tables", "images" and links to other pages. The page also needs a better "Text Editor" so the users can add complex text.
* The page also has to be made public or anonymous based on the content for the admin to decide if its suitable for publishing
* The images that can be uploaded have to be located on a portion of the page that is mostly on the right side and hence then can collapse below the text to prevent misplacement when the user is viewing the page.

**Task 3 (30 minutes)**

Do you see any improvements that could’ve been done to [the code attached](https://getadigital.com/globalassets/career/refactoring-task.zip)? If you do, take some notes on how you would refactor, clean it up and make, the code more maintanble  (the code won’t compile – this is more a mental exercise).

We know that the attached code is written in web forms. None of our projects are using web forms any more (and haven't for many years). This is just to have some old legacy code to refactor and talk about improvements.

**CodeRefactoring**

* **ApplicationForm.aspx page**
  + Remove inpage css styles and add a new CSS stylesheet as a link

- It will reduce the amount of code that is run when the page is loaded

* Can create a new script file for the jQuery code and run the script file in the background via a link

- It will reduce the amount of code that is run when the page is loaded

* Can instead use KnockOutJS and get rid of old jQuery code.

* Replace HTML4 ( <div><div> ) structure that is defined in the page with HTML5 <table> structure that will define rows, columns, header, footer etc

* **ApplicationForm.aspx.cs**
  + Can get rid of all the comments in the code and rename the variables such that they mean what they are doing
  + Can get rid of #region #endregion as it is not good practise.
  + Replace PageOnLoad event in line 20 with "public static void Main(string[] args) {} and run the code from there
  + Need to move public class ContactPerson to top of the code

* Replace "protected string[] countyList" with a list of type String List<string>

* Add a new item to contactPersonList, initiation can be done on the same line number 16 instead of 2 lines like

Protected List<ContactPerson> contactPersonList = new List<ContactPerson>();

* Can exclude "new ContactPerson" from all the lines between line 400 and 456 and just add the object directly

contactPersonList.Add("Sørfold", "Nordland", "Kjell.Stokbakken@Legacy.com");

* Populating the dropdownlist should be done mainly from the database using data stored in a table. That will eliminate 56 lines of code in the backend code

* Have to use "using statements" for retrieving the CurrentLanguage name and code instead of using functions

Using EPiServer.Globalization.ContentLanguage.PreferredCulture.

Here we can retrieve the name into a static string easily

* Can replace method GetEmailForMunicipality with a Dictionary method, thus, preventing running the foreach loop. Can also move the PopulateContactPersonList() method to the top of the page.

* In the BuildEmailContent() method, instead of using stringBuilder, can make use of "MailDefinitionClass" provided by C#.
* Can rewrite the SendEmail function to return valid "Success" message or exception message(which is not getting returned in the catch statement). Can add a "finally" statement to the try-catch-loop to return the email sent status.

Can include the "if(ok)" as a else statment in the if loop above.

* P.S: there is so much more I can do to make the code more better and into a newly updated version. But these are the things that came on the top of my head **in 30 minutes**